Working on the HPC2 cluster

Shell (Bash/sh) scripting

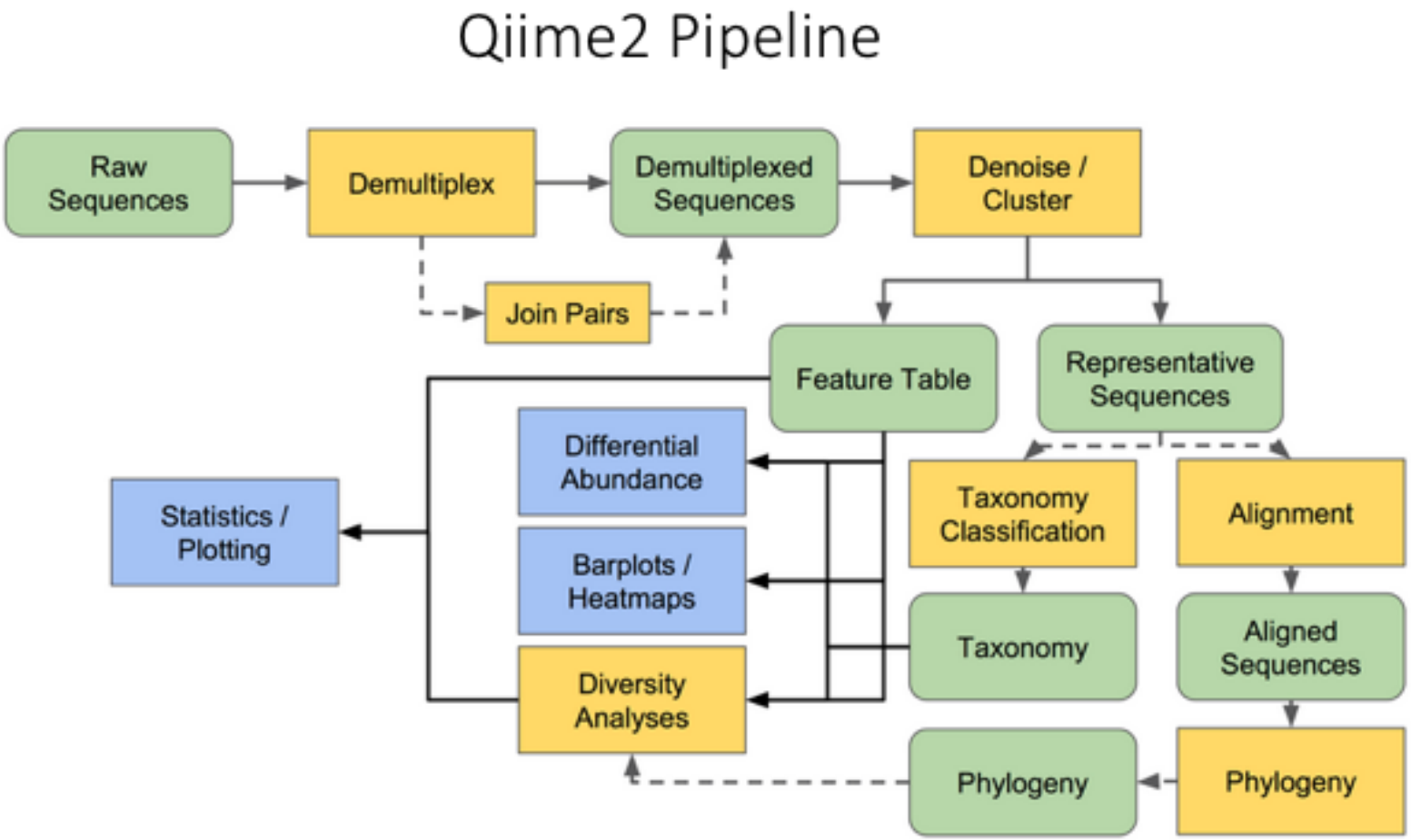
Andrew Ndhlovu

Project SeaStore
Department of Botany and Zoology
Stellenbosch University
https://github.com/PiscatorX/RNA-Seq-devs

July 20, 2023

Bioinformatic pipelines and workflow systems

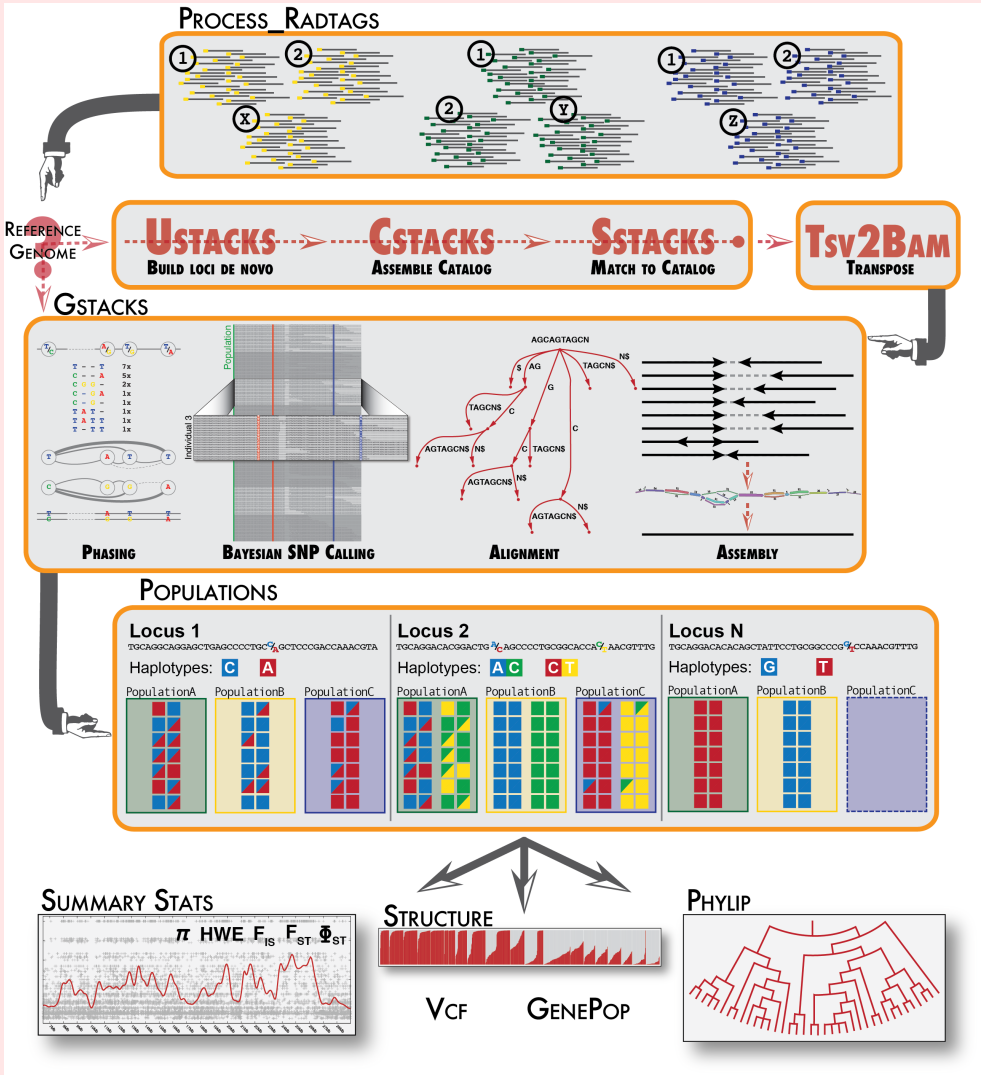
How do we run (and re-run) analyses in way that saves time?



https://otagoedna.github.io/edna_workshop_june2021/chapters/04_intro_to_qiime.html

Bioinformatic pipelines and workflow systems

We also need to try different options or tools



https://catchenlab.life.illinois.edu/stacks/manual/

How do we get started

https://doi.org/10.1371/journal.pcbi.1008645

Building a bioinformatic pipeline

What is a pipeline

A workflow consisting of a variety of steps (commands) and/or tools that process a given set of inputs to create the desired output files.

Syntax	Paradigm	Interaction	Example	Ease of Development	Ease of Use	Performance
Implicit	Convention	CLI	Snakemake, Nextflow, BigDataScript	★★★½	★★★★½	★★★
Explicit	Convention	CLI	Ruffus, bpipe	★★★★	★★★★	★★★
Explicit	Configuration	CLI	Pegasus	★★½	★★★	★★★★
Explicit	Class	CLI	Queue, Toil	★★½	★★★	★★★★★
Implicit	Class	CLI	Luigi	★★★	★★★½	★★★★★
Explicit	Configuration	Open Source Server Workbench	Galaxy, Taverna	★★★	★★★★½	★★★
Explicit	Configuration	Commercial Cloud Workbench	DNAnexus, SevenBridges	★★½	★★★★★	★★★★½
Explicit	Configuration	Open Source Cloud API	Arvados, Agave	★★★	★★★★	★★★★½

https://doi.org/10.1093/bib/bbw020

Most used advanced work flow management systems

snakemake
A framework for reproducible data analysis

COMMON
WORKFLOW
LANGUAGE

Criteria	Snakemake	cwltool	Toil	Nextflow
Installation method used	conda	Python pip	Python pip	conda
Number of RiboViz processing steps implemented	14	3	3	5
Time (person-days) to implement steps	1	1	1[1]	2
Iteration over multiple samples	Yes	See note[2]	See note[2]	Yes
Error recovery strategies	Yes ("keep going" parameter)	See note[2]	See note[2]	Yes (step-specific error strategies)
Aggregation of sample-specific results	Yes	See note[2]	See note[2]	Yes
Conditional execution of steps	Yes (step-specific Python conditions)	No	No[3]	Yes (step-specific "when" clauses)
Step-specific log files	Yes (must be captured explicitly by each step)	See note[2]	See note[2]	Yes (automatically captured)
YAML configuration files	Yes	See note[2]	See note[2]	Yes
"dry run" option	Yes	See note[2]	See note[2]	No (but configuration validation can be implemented)
Output a rerunnable bash script to see the commands that were actually executed	No (outputs a summary file with bash commands)	See note[2]	See note[2]	No (outputs step-specific bash scripts)
Execution within containers, HPC systems, and cloud	Yes	Containers only	Yes	Yes

`https://doi.org/10.1371/journal.pcbi.1008622`

Using a shell script as pipeline

A shell script is a series of commands and instructions written in a scripting language usually run on a UNIX/LINUX operating system

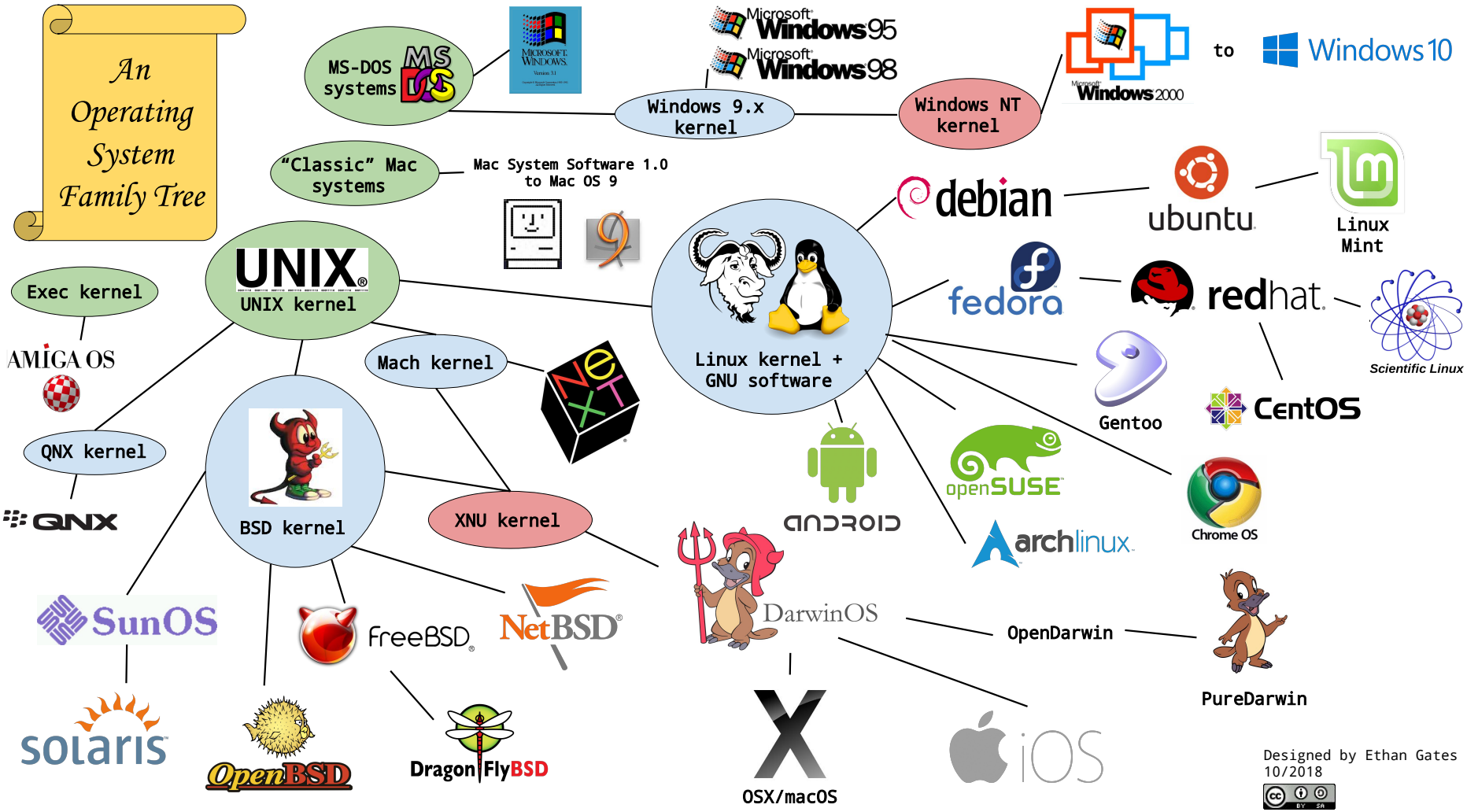
Save the commands that we run in a file

- Simplest pipeline to implement
- Easiest to learn for beginners
- Usually written in the Unix shell scripting language (Bash/sh)
- Variables and conditional logic can used for flexibility

Simplicity comes at a cost

- Script pipelines (usually) cannot be continued
- Challenging to implement parallel running of tasks
- Become cumbersome and challenging as they become longer
- Still requires some understanding of Linux commands and File system

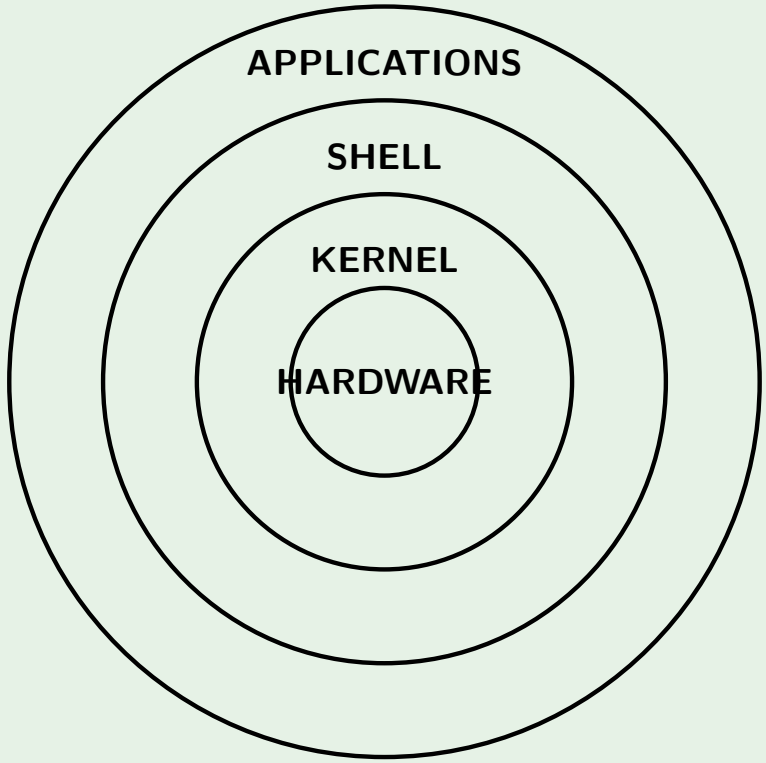
Where to run your pipeline: Many Operating systems



https://github.com/EG-tech/digipres-posters

What is the shell?

Architecture of a Operating System (OS)



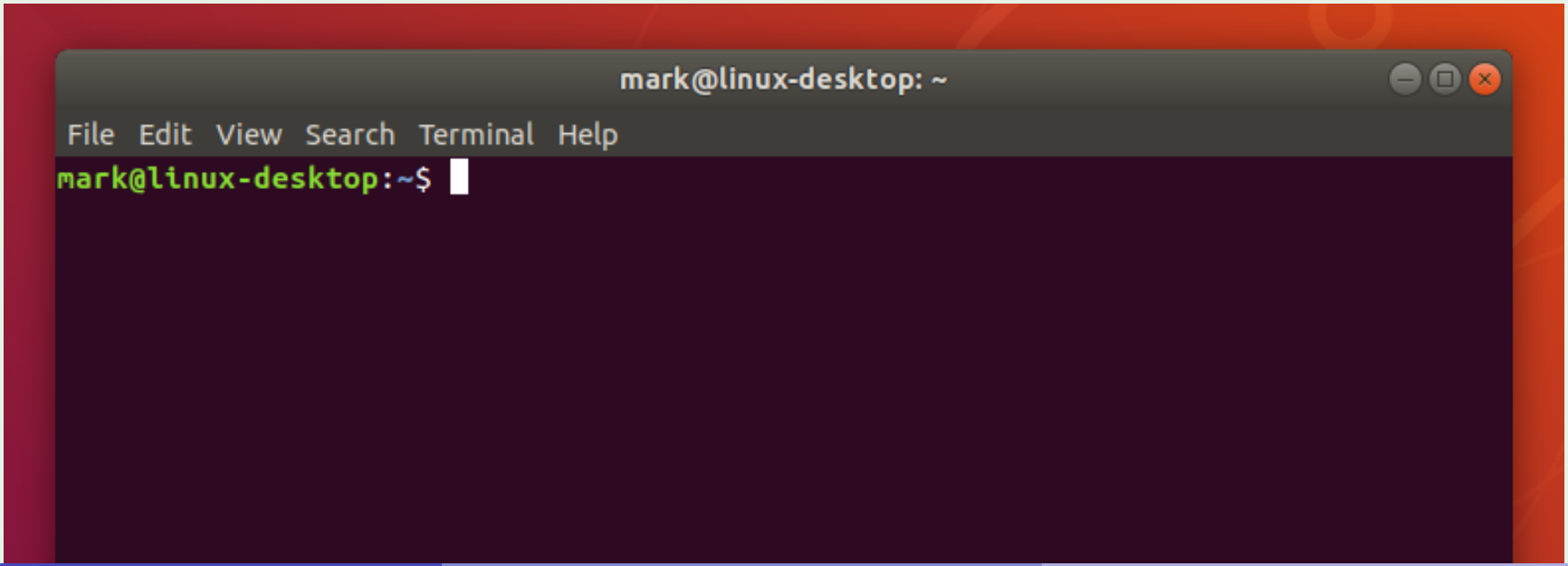
- Shell provides an interface to the OS and services kernel.
- It is a layer of abstraction to send instructions to the OS/kernel
- Kernel sends instructions to the hardware i.e CPU, RAM etc

Introduction to Bash/sh shell scripting

Unix shell is both a command-line interface (CLI) and a scripting language, allowing automating repetitive tasks

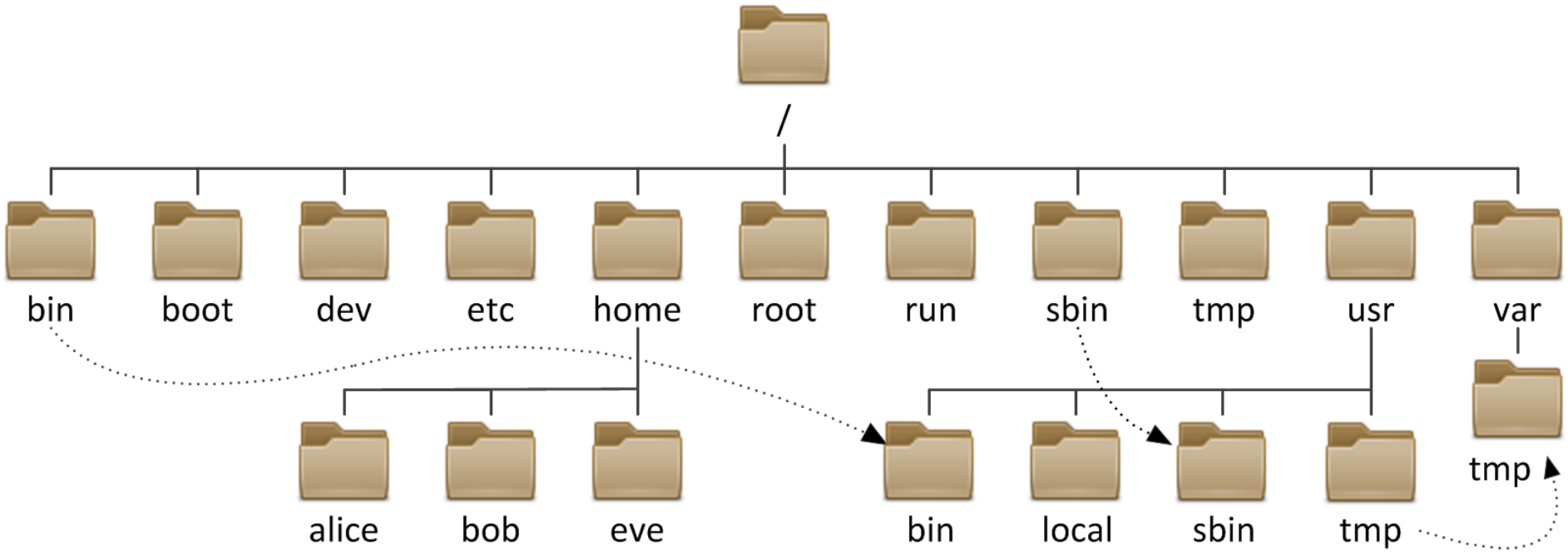
- Many shells e.g. Bourne shell (sh), C shell (csh) and zsh on Mac
- Bash (Bourne Again SHell) is the default shell on Linux terminal
- On the command line bash is indicated by the dolar sign prompt $
- Prompt usually preceded by username, machine name and current directory

`https://ubuntu.com/tutorials/command-line-for-beginners#3-opening-a-terminal`



Getting started: Typical Linux directory structure

https://help.ubuntu.com/community/LinuxFilesystemTreeOverview



https://www.serverkaka.com/2018/01/key-locations-in-linux-file-system_21.html

File system navigation

But first: What is a path?

A path in Unix is the location or address of a file or directory within the Unix file system

- Absolute path: full address starting from root "/" e.g: /home/andhlovu/mydata/output
- Relative path: not a full address on relative to where you are e.g: mydata/output

Special directory name

Name	Description
~	(tilde) means the user's home directory, usually /home/user-name
.	(dot) means the current directory you're in
..	(dot dot) means the parent directory (one level above) of the current directory you're in. For example, if you're in foo/bar/, . will represent bar/, `..` `will represent foo/`.

File system navigation commands

Command	Description
`pwd`	Print name of current/working directory. Print the full filename of the current working directory.
`cd [directory]`	Change the working directory to directory, can be absolute or relative path.
`ls`	List directory contents. List information about the files in the current directory by default.
`ls -l`	List detailed directory contents. - is called a flag. flag/minus sign is used to pass options to command
`ls -l -s`	List detailed directory contents with file sizes
`ls -ls`	short version on `ls -l -s`

Need to check the options of command: seek `--help` or check the `man (manual) pages`

Command	Description
`ls --help`	Print all the usage and options of the `ls command. No need to try to remember every option.`
`man ls`	Provides full name, synopsis, description, options and examples of the `ls command.`

Also make use Google, the more you search for commands, your results get better over time when you look up

File management

Copying Files: Command Syntax

```
cp source_file destination_file
```

Example

```
cp file.txt /path/to/new/location/
```

Renaming Files: Command Syntax

```
mv old_file_name new_file_name
```

Example

```
mv file.txt new_file.txt
```

File management

Moving Files: Command Syntax

```
mv source_file /path/to/new/location/
```

Example

```
mv file.txt /path/to/new/location/
```

Deleting Files: Command Syntax

```
rm file_name
```

Example

```
rm file.txt
```

Important Tips

- Double-check the destination path to avoid accidental overwrites or data loss.
- Use the `mv` command with caution, as there is no undo option.